

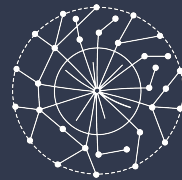
Pushing Matrix-Vector Multiplication Performance on AMD AI Engines for Low-Latency Edge Inference

Fast Machine Learning for Science Conference

ETH Zurich September 1-5th, 2025



EP-SFT
Software Frameworks and Tools



NextGen

Dimitrios Danopoulos, Roope Oskari Niemi, Enrico Lupi, Anastasiia Petrovych, Sebastian Dittmeier, Michael Kagan, Vladimir Lončar

Motivation and Background

Matrix-vector multiplication (GEMV) is common in AI workloads (e.g., CNNs, MLPs)

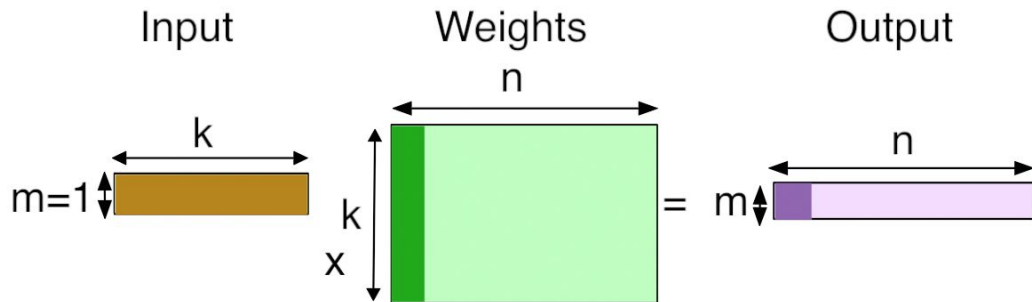
But unlike General Matrix Multiply (GEMM), GEMV is harder to optimize
→ **low data reuse** and **memory-bound** behavior make it less efficient.

On AIE-ML architecture, GEMV (e.g., int8) delivers only ~ 1 OP per byte — **can't fully utilize compute units**.

For context: in an Apple M4 GPU

- GEMM: 14188 GFLOPs (FP16)
- GEMV: 187 GFLOPs (FP16)

→ Only **1.3%** of peak performance



These inefficiencies would matter in ultra-low-latency environments, such as the **CERN Level-1 (L1) trigger system**.

Motivation and Background

AMD AI Engines (AIEs) can offer a promising low-latency alternative (for example to CERN's traditionally used FPGAs)

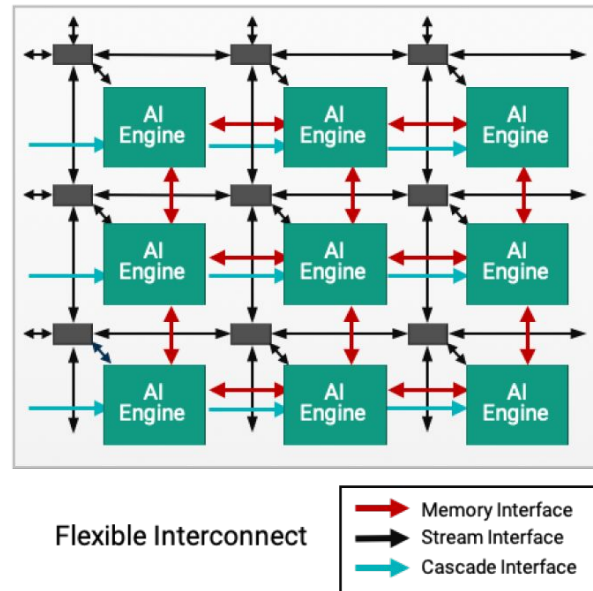
With higher compute density and on-chip memory, AIEs may better handle medium to large neural networks compared to FPGAs.

Regarding CERN, they provide the necessary high speed I/O interfaces for the trigger.

Experiments like ATLAS and CMS are actively exploring AIEs.

However, major barriers exist:

- Difficulty in designing generic & scalable designs
- Development is still low-level and cumbersome, with limited toolchain support



About the architecture

2D array of VLIW vector processors with shared memory and direct interconnects

Programmed using C/C++ with support of vector intrinsics for perf. optimization

Can scale workloads by partitioning them across the AI tiles

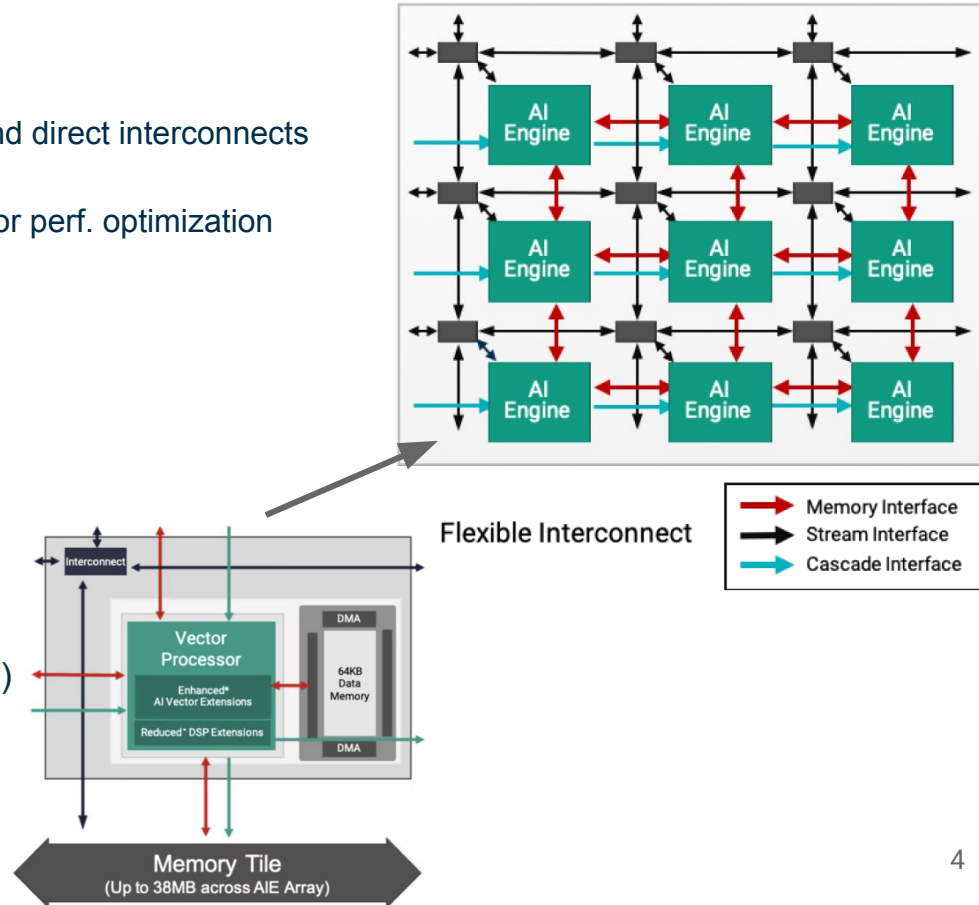
We will target a Versal AI Edge series device

VE2802

Built on the 2nd generation of AIE architecture (AIE-ML)

Increased compute and memory vs 1st gen devices

New dedicated Memory tiles support



Kernel Optimization:

- Choose I/O interfaces based on dataflow needs.
- Optimize kernel code via software pipelining and SIMD usage.
- Use L1 memory effectively: rearrange data, enable double buffering.
- Avoid memory bank conflicts to maximize tile memory throughput.
- Add ReLU + bias support to enable a dense layer equivalent for NNs

AIE array optimization:

- Partition workload across tiles horizontally and vertically.
- Scale GEMV operations via data tiling and parallel kernel execution.

Graph level optimization:

- Leverage MEM tiles for input/output reordering and inter-graph communication.
- Use graph-level pipelining to overlap layers and reduce end-to-end latency.
- Compact graph layout via manual placement to ensure predictable behavior.

Place buffers to **avoid bank conflicts** and maintain predictable tile placement.

→ See picture on the right. 64 KB local memory (4×16 KB banks)

Aligned, wide vector load/store

Use 256-bit aligned accesses across banks to maximize memory throughput per cycle.

Double buffering for inputs

Ping-pong buffers can hide memory latency by overlapping data transfers with computation.

Use SIMD intrinsics and software pipelining

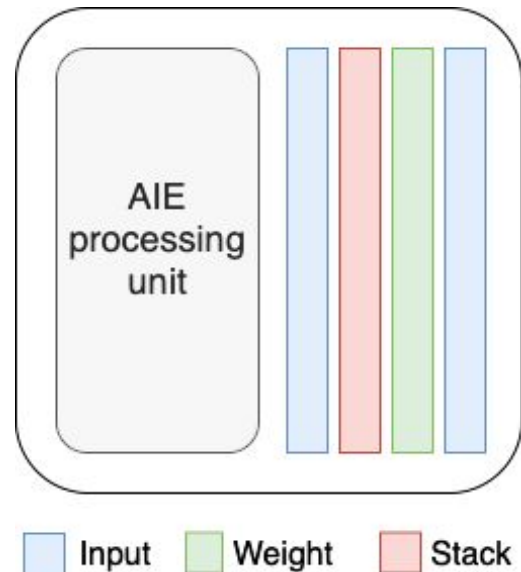
→ data-level parallelism via vector operations (SIMD)

→ instruction-level parallelism (ILP) through VLIW architecture

Keep MAC units busy via batching

Process input vectors in batches (e.g., $B \times N$), reusing the same weight matrix across multiple vectors

Batching allows full utilization of all accumulator lanes, whereas single-vector GEMV underutilizes them.



AIE array strategy

Distribute the workload across tiles **both horizontally and vertically** to fully utilize the 2D AIE array.

West-to-East cascading for computing dot-product accumulations

South-to-North cascading for computing a slice of the output vector

The shared input vector is **broadcast upward** from the memory tile

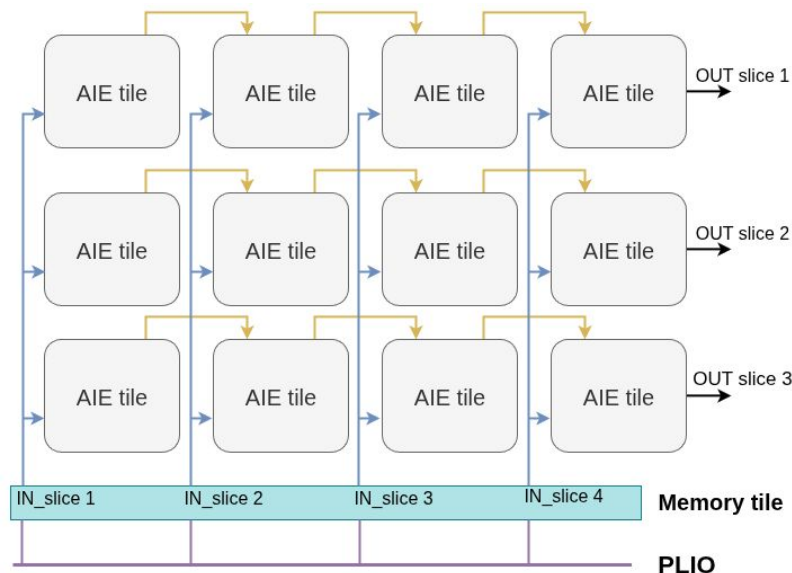
Support scalable, low-latency matrix-vector computation by adapting cascade configurations to the layer shape:

More input features

→ use wider cascade rows

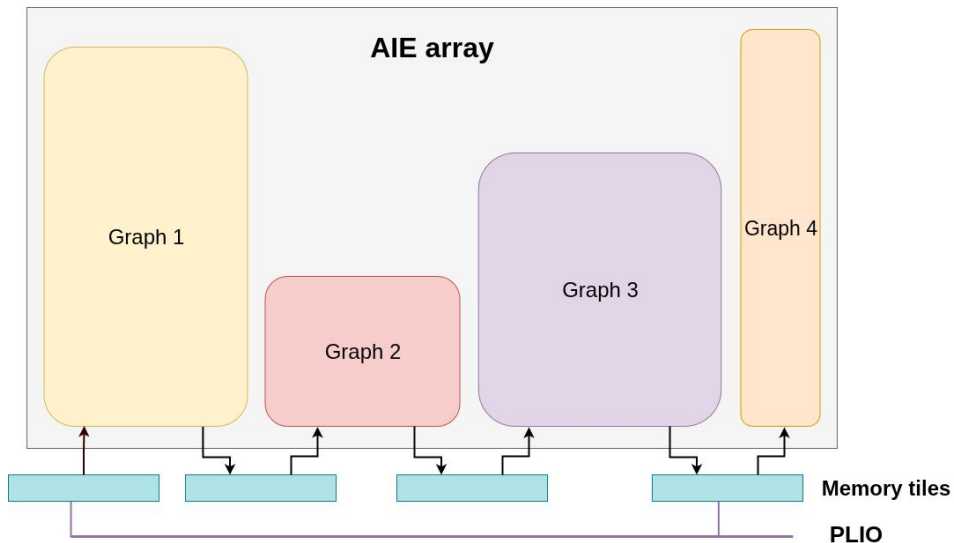
More output neurons

→ replicate cascade rows



Layer-to-layer communication via Memory tiles

- MEM tiles act as a double-buffered stage to pass intermediate results between graphs (layers).
- Enable overlap of execution between layers
- Accumulate or reorganize intermediate results before feeding them into the next layer



Performance comparison of GEMV across different configurations

- Batching improves MAC utilization and throughput.
- Our AIE kernel implementation achieves close to theoretical peak performance

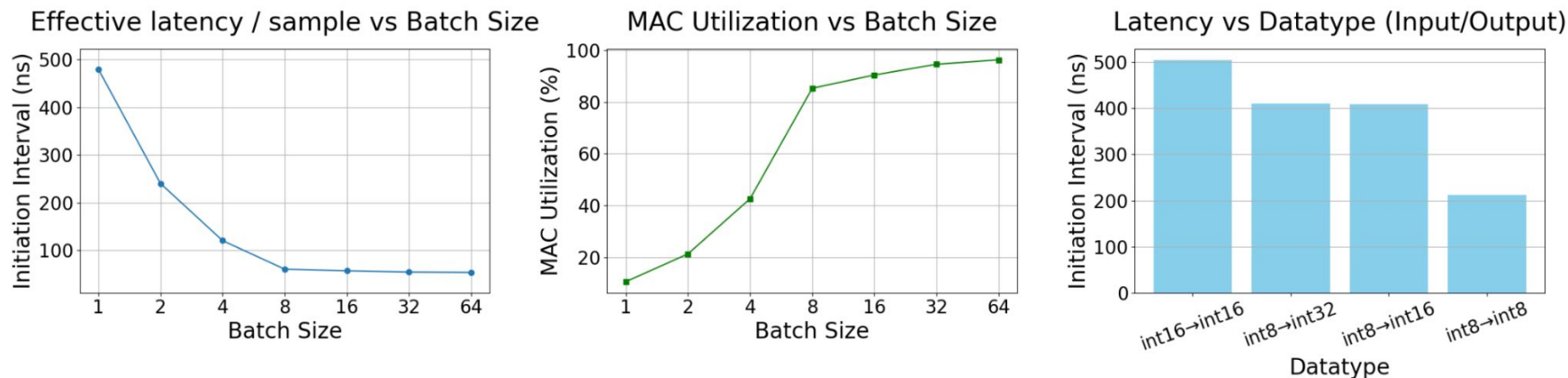


Fig.1 & Fig.2 : single AIE tile, 256-input × 64-output layer

Fig.3 : single AIE tile, 128-input × 32-output layer, batch size = 8, Input/weight datatypes are assumed equal.

Performance comparison of GEMV across different dataset configurations

- Actual throughput curve diverges from the expected linear scaling as tile count increases
- This is expected (congested paths, communication overhead, etc.)

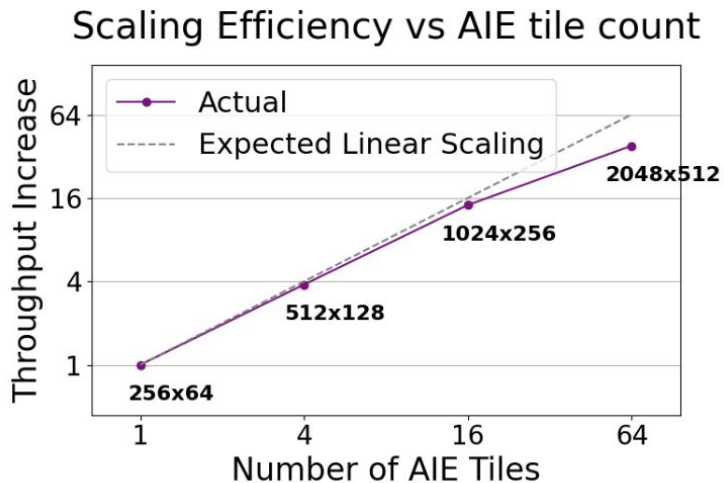


Fig.1 & Fig. 2 : int8 datatype used for inputs/weights

Connecting multiple layers together

3 layer MLP measuring end to end performance

Compilation time: 13min



AIE array utilization

MLP configuration

Input: (_, 2048)

Layer 1: Linear(2048 \rightarrow 512) + Bias \rightarrow ReLU

Layer 2: Linear(512 \rightarrow 128) + Bias \rightarrow ReLU

Layer 3: Linear(128 \rightarrow 16) + Bias \rightarrow ReLU

Output: (_, 16)

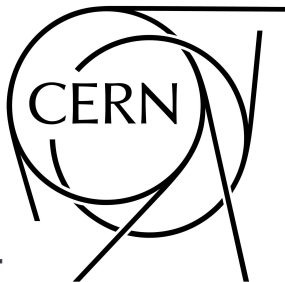
Results:

Iteration interval (II) : 856.1 ns

With batch size 64:

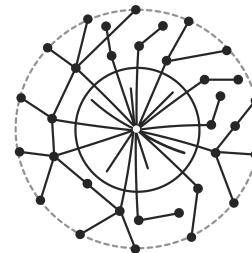
Iteration interval (II) : 6618 ns

Derived II per sample : 103 ns



EP-SFT

Software Frameworks and Tools



NextGen

Thank you!

This work has been [partially] funded by the Eric & Wendy Schmidt Fund for Strategic Innovation through the CERN Next Generation Triggers project under grant agreement number SIF-2023-004.